

APPENDIX A

Sample Code for First Preferred Embodiment

```
/*=====
```

Method1 sample code:

5

NOTE: This code is intended to demonstrate the key points of the implementation for method 1. It is intended for clarity and simplicity, so it has not been optimized.

10

The code is written in C with two C++ extensions:

- * C++ style comments (everything from "//" to the end of the line is a comment)
- * Variables can be declared anywhere in a function, not just at the start of a scope.

15

This code assumes:

- * An OpenGL context to display the pixels has been created and is active
- 20 * The GL renderer supports the GL_TEXTURE_RECTANGLE_EXT extension - non power of 2 pixels. The method would work without the extension, but would not be as optimal or simple.
- 25 * The size of the out-of-order pixel data is stored in sPixelDataRect.
- * The size of each pixel is stored in sBytesPerPixel. This code assumes a 2 or 4 byte pixel. The method will work with 1 byte pixels, but the implementation is more complicated.
- 30 * A 2D texture, the same size as the out-of-order

pixel data has been created and is bound to the
id stored in sTextureID.

Written by: Mick Foley (mickf)

5

Copyright: 2003 Microsoft

=====*/

// header files that define the OpenGL data types,
10 // values and functions

#include <gl.h> // the OpenGL header
#include <glxt.h> // OpenGL extensions

15 // local type definition

typedef struct struct_tRect
{
 long fTop;
20 long fLeft;
 long fBottom;
 long fRight;
} tRect;

25 // static data - see the notes for more info

static void* sPixelData_BaseAddress;
static tRect sPixelDataRect;
static long sBytesPerPixel;
30 static int sTextureID;

// the code

```
void Method1_DrawPixels( void )
{
    long stripWidth = 8 / sBytesPerPixel;
    long numRectangles =
5      ( sPixelDataRect.fRight
        - sPixelDataRect.fLeft )
    / stripWidth;
    long iCounter;

10    // make sure that texturing is on
    // and we have the correct texture set
    glEnable(GL_TEXTURE_RECTANGLE_EXT);
    glBindTexture(GL_TEXTURE_RECTANGLE_EXT, 1);

15    // update the texture with the data
    // from the emulator VRAM
    if ( sBytesPerPixel == 2 )
    {
        glTexSubImage2D( GL_TEXTURE_RECTANGLE_EXT,
20          0,
            sPixelDataRect.fLeft,
            sPixelDataRect.fTop,
            sPixelDataRect.fRight -
                sPixelDataRect.fLeft,
25          sPixelDataRect.fBottom -
            sPixelDataRect.fTop,
            GL_RGB,
            GL_UNSIGNED_SHORT_5_6_5,
            sPixelData_BaseAddress );
30    }
    else
    {
        glTexSubImage2D(GL_TEXTURE_RECTANGLE_EXT,
```

```

    0,
    sPixelDataRect.fLeft,
    sPixelDataRect.fTop,
    sPixelDataRect.fRight -
5      sPixelDataRect.fLeft,
    sPixelDataRect.fBottom -
      sPixelDataRect.fRight,
    GL_BGRA,
    GL_UNSIGNED_INT_8_8_8_8_REV,
10    sPixelData_BaseAddress);
}

// draw the rectangles

15 // start issuing rectangle commands
glBegin( GL_QUADS );

// issue a rectangle for each strip
// with the horizontal texture coords reversed
20
for ( iCounter = 0; iCounter < numRectangles;
      iCounter++ )
{
    long stripTop =      sPixelDataRect.fTop;
25    long stripLeft =    sPixelDataRect.fLeft
        + ( iCounter * stripWidth );
    long stripBottom =   sPixelDataRect.fBottom;
    long stripRight =    stripLeft + stripWidth;

30    // upper left vertex -
    // upper right texture coord
    glTexCoord2i( stripRight, stripTop );
    glVertex2i( stripLeft, stripTop );
}
```

```

    // upper right vertex -
    // upper left texture coord
    glTexCoord2i( stripLeft, stripTop );
5    glVertex2i( stripRight, stripTop );

    // lower right vertex -
    // lower left texture coord
    glTexCoord2i( stripLeft, stripBottom );
10    glVertex2i( stripRight, stripBottom );

    // lower left vertex -
    // lower right texture coord
    glTexCoord2i( stripRight, stripBottom );
15    glVertex2i( stripLeft, stripBottom );
}

    // done with the rectangle draws
    glEnd();
20

    // finished with all the commands
    glFlush();
}
```